Inteligência Artifical Aplicada à Navegação Autônoma de Robôs

Thiago da Silva Almeida, Nathanael Oliveira Vasconcelos,
Fabricio de Oliveira Fonseca e Alcides Xavier Benicasa
Departamento de Sistemas de Informação - DSI
Universidade Federal de Sergipe - UFS
Itabaiana-SE, Brasil
e-mail: thiago7a@gmail.com; nathan-vasconcelos@hotmail.com;
tr3mere@hotmail.com; alcides@ufs.br

***Resumo*—Este artigo possui como objeto apresentar o trabalho voltado ao desenvolvimento e utilização de algoritmos inteligentes para o controle da navegação do robô. O principal objetivo foi desenvolver mecanismos capazes de guiar o robô, partindo de dois pontos previamente definidos, sendo o primeiro a posição inicial de partida do robô e, o segundo, o local para o qual o robô deverá se deslocar, considerando para isso um ambiente complexo, composto de várias salas. Os métodos inteligentes implementados para a navegação serão validados e comparados ao final.**

***Keywords*-Controle de navegação, robótica, inteligência artificial, métodos de busca;**

I. Introdução

A navegação autônoma de robôs é um dos temas mais pesquisados na área de computação nos últimos tempos, pois se torna cada vez mais importante ter uma máquina capaz de chegar a lugares onde o ser humano não chega e que pense de maneira semelhante a um ser humano.

Apesar da volumosa quantidade de pesquisas sobre o assunto, o tema ainda é desafiador, pois fazer com que um robô analise um ambiente e o percorra de maneira inteligente sem esbarrar em nada não é uma tarefa trivial. Ainda assim, a pesquisa nessa área pode ser de grande proveito para a comunidade, uma vez que o uso de robôs móveis pode ser feito nas mais diversas áreas, como por exemplo: limpeza de tubulações, controle de incêndios, vasculhamento de grandes áreas, pois estas, apesar de serem tarefas que podem ser executadas por humanos, tem uma natureza repetitiva de padrões e/ou são perigosas.

Sendo assim, o objetivo desta pesquisa é apresentar maneiras de guiar de forma inteligente um robô móvel. Neste trabalho usaremos algoritmos de Inteligência Artificial pressupondo que o robô já conhece o ambiente em que se encontra, dessa forma os algoritmos serão a base de tomada de decisão e movimentação do robô para que este possa se locomover de forma harmoniosa no ambiente.

Para visualizar os resultados de forma mais concreta utilizamos uma ferramenta de simulação, e inserimos o robô em um ambiente definido por nós. Abordaremos neste estudo todos os conceitos necessários para o entendimento e desenvolvimento da pesquisa, a importância do uso de ferramentas de simulação, o funcionamento específico da ferramenta de simulação utilizada neste caso e por fim a aplicação de algoritmos de inteligência artificial, seguido de uma avaliação dos resultados.

II. Revisão da Literatura

É perceptível o crescimento de estudos na área de robótica, ainda assim o tema exige muito do pesquisador, de acordo com WOLF and Trindade Jr. (2009), a pesquisa e desenvolvimento em robótica requerem conhecimentos em diversas áreas da engenharia como, por exemplo, mecânica, elétrica e computação. Para pesquisas envolvendo robôs móveis inteligentes deve-se também considerar a extração de informações do ambiente de domínio, utilizando-os para realizar determinada tarefa.

RUSSELL (2003) define robô como sendo um agente físico que executa tarefas manipulando o mundo físico. Para tanto eles precisam de 'efetuadores', que nada mais são que o meio pelo qual o robô atua e exerce força sobre o ambiente. Além disso o robô pode possuir sensores a fim de poder perceber o ambiente em que se situa.

RUSSELL (2003) cita três tipos de robôs na sua pesquisa: manipuladores, robôs móveis e os robôs híbridos. Segundo ele os manipuladores, ou braços robôs, estão fisicamente ancorados (ou fixos) a seu local de trabalho, por exemplo, em uma linha de montagem industrial ou na estação espacial internacional. O outro tipo de robô citado por RUSSELL (2003) é o tipo que trabalharemos neste estudo, o robô móvel, que se desloca por seu ambiente usando rodas, pernas ou mecanismos semelhantes. Eles foram projetados para uso na entrega de alimentos em hospitais, para mover contêiners em docas de carga e tarefas semelhantes. O terceiro tipo, robô híbrido, trata-se de uma junção dos outros dois tipos, temos como exemplo o robô humanóide.

RIBEIRO et al. (2001) explica o funcionamento do robô inteligente da seguinte maneira:

- Perceber a situação do ambiente através de seus sensores;
- Determinar a ação (ou ações) a executar, eventualmente raciocinando para interpretar percepções e resolver problemas;

- Executar uma ação, através de seus atuadores, que afeta as condições do ambiente e;
- Produzir novas situações.

Sendo assim, um robô inteligente alcança o sucesso esperado na execução da tarefa a ele designada quando, por exemplo, alcança um local previamente estabelecido desviando dos obstáculos, evitando assim colisões. Porém, isto é mais complexo do que se imagina, uma vez que o mundo real apresenta uma natureza ruidosa, imprevisível e dinâmica (RIBEIRO et al., 2001).

Imprevistos ocorridos no ambiente necessitam de reações rápidas, por esse motivo um dos grandes desafios da robótica é justamente como integrar as informações vindas de todos estes sensores, de modo a gerar comandos e controlar os diferentes dispositivos de atuação do robô, garantindo que a tarefa seja executada de modo correto e sem colocar em risco, tanto o robô, quanto àqueles que o cercam (WOLF and Trindade Jr., 2009).

A. Ambiente de Simulação - Player e Stage

Para aplicarmos os algoritmos inteligentes houve a necessidade de utilizar um ambiente virtual de simulação de robôs, isso nos proporciona diversas vantagens, as quais WOLF and Trindade Jr. (2009) demonstram em seu trabalho: economia de tempo, pois podemos realizar um maior número de experimentos através de simulação; evitar gastos, que além de não necessitar comprar ou fazer um robô inicialmente, evitando danos, devido, por exemplo, a fortes colisões, acionamento indevido dos motores, etc.

A ferramenta e plataforma escolhida foi o *Player* e *Stage*, por se adaptarem melhor ao estudo de robôs simulado, também conhecidos como *Player/Stage*, que juntos fazem a simulação do robô, tornando as pesquisas mais rápidas e com menor custo.

Trata-se de um sistema de código aberto e de livre distribuição (compatível com o Linux), possui a colaboração de diversos pesquisadores das mais diversas instituições, sendo amplamente utilizado por universidades, institutos de pesquisa e empresas em diversos países. De acordo com OWEN (2010), o *Player/Stage* funciona como uma camada de abstração de hardware. Isso significa que ele comunica-se com os $bits$ de hardware do robô (como uma garra ou uma câmera, por exemplo) e permite ao usuário controlá-los com seu código, ou seja, não precisa se preocupar como as várias partes do trabalho do robô. O *Stage* é um plugin para o *Player* que “ouve” o que ele está dizendo para executar, e retorna novas instruções para a simulação do robô. Ele também simula os dados do sensor e envia para o *Player*, o qual por sua vez faz com que os dados dos sensores disponíveis retornem para o código.

A figura 1 apresenta o fluxo de comunicação entre o *Player/Stage* e o simulador, iniciando no código e finalizando na aplicação.

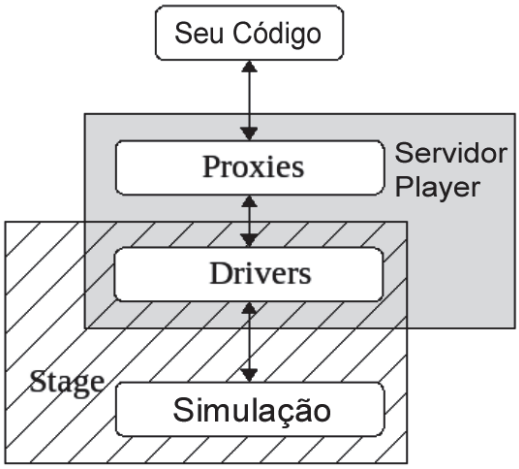

Figura 1. Estrutura de comunicação entre o Player/Stage e o simulador.

O ambiente de simulação *Player/Stage* possui métodos que simulam de forma eficiente um determinado ambiente definido ou elaborado pelo desenvolvedor, ainda que não possa garantir seu comportamento de forma idêntica em um ambiente real, permite que o usuário construa seu robô e/ou mundo da forma que preferir.

B. Inteligência Artificial e Algoritmos de Busca

Para atingir o objetivo deste trabalho tornou-se indispensável o uso da teoria da Inteligência Artificial.

De acordo com LUGER (2004), IA é "um campo de estudo jovem e promissor, cujo interesse principal é encontrar um modo efetivo de entender e aplicar técnicas inteligentes para solução de problemas, planejamento e habilidades de comunicação a uma ampla gama de problemas práticos", ou seja, é uma área de pesquisa que se dedica em buscar métodos ou dispositivos computacionais que possuam ou simulem a capacidade racional de resolver problemas, especificamente o que chamamos de pensar.

Em seus estudos, RUSSELL (2003) define agente como "tudo o que pode ser considerado capaz de perceber seu ambiente por meio de sensores e de agir sobre esse ambiente por intermédio de atuadores". Ele ainda define um agente racional da sequinte forma: "para cada sequência de percepções possível um agente racional deve selecionar uma ação que se espera venha a maximizar sua medida de desempenho, dada a evidência fornecida pela sequência de percepções e por qualquer conhecimento interno do agente", ou seja, os dados que são recebidos pelo agente devem gerar um uma saída correta e otimizada, na nossa pesquisa tentamos encontrar a melhor saída correta com o menor custo possível. Tendo isso em mente, a medida de desempenho do nosso agente, ou ‘critério para medir o sucesso do seu comportamento’(RUSSELL, 2003), é encontrar o caminho

mais curto com o menor custo computacional para encontrá-lo, neste caso específico a quantidade de nós expandidos.

Para que o agente apresente determinado comportamento é necessário que ele atue sobre determinado ambiente, o ambiente que definimos é composto por vinte e duas salas, e o nosso agente deve atuar no ambiente locomovendo-se de uma sala origem para uma sala objetivo, sendo que a sequência de salas que devem ser percorridas entre a origem e o objetivo deve ser encontrada pelo agente.

Segundo RUSSELL (2003), um agente com várias opções imediatas de valor desconhecido decidindo o que fazer examinando primeiro diferentes sequências de ações possíveis que levam a estados de valor conhecido, e depois escolhendo a melhor sequência, o processo da procura desta sequência é chamado de busca.

Este trabalho concentrou-se em algoritmos de busca para o caminhamento autônomo do robô por um ambiente previamente conhecido. Dentre os métodos de busca cega e com informações encontrados na literatura os seguintes algoritmos de busca serão utilizados: profundidade, extensão ou largura, custo uniforme, gulosa pela melhor escolha e a forma mais amplamente conhecida da busca pela melhor a busca A*.

Inicialmente foi necessário definirmos a heurística a ser utilizada, de forma que, para fins de comparação entre os métodos propostos, a heurística, quanto utilizada, será a mesma entre os algoritmos implementados. A informação utilizada para tomada de decisão, de acordo com a sistemática de cada algoritmo, foi a distância euclidiana, descrita na Equação 1.

$$D = \sqrt{(X - x)^2 - (Y - y)^2} \tag{1}$$

onde X e Y representam as coordenadas referente ao ponto inicial e os valores de x e y representam as coordenadas do ponto final.

A busca em profundidade e a busca em largura possuem algo em comum, ambos não possuem nenhuma heurística referente ao cenário, ou seja, não possuem nenhuma informação sobre as distâncias de um ponto ao outro, por isso são conhecidas como buscas cegas, levando em consideração apenas a sequência de nós vizinhos para dar sequência ao processo de busca, no entanto, é neste parâmetro que ambos se tornam distintos, o algoritmo de profundidade navega entre os filhos, antes de navegar entre os nós vizinhos, portanto, é um algoritmo que, segundo RUSSELL (2003), quando um estado é examinado, todos os seus filhos e os descendentes são examinados antes de qualquer um de seus irmãos, ou seja, essa busca avança se aprofundando no espaço do estados sempre que possível. A busca em largura expande todos os filhos do nó atual, se considerarmos um determinado estado como sendo um nó de uma árvore. Assim, a busca em largura normalmente expande mais nós que a busca em profundidade, porém apresenta uma possibilidade maior de encontrar o melhor caminho, sendo que, se consirarmos o custo de cada nó for igual a 1, a busca em largura sempre achará o melhor caminho.

Os demais algoritmos usados nesta pesquisa, custo uniforme, gulosa pela melhor escolha e A*(A estrela), possuem alguma heurística, ou seja, alguma informação, diferentemente dos citados a cima. O custo uniforme leva sua consideração heurística na expansão do nó não expandido de custo mais baixo, considerando dessa forma apenas a distância percorrida e não a distancia até o objetivo, caso este, totalmente diferente da gulosa pela melhor escolha, que leva em consideração apenas a distância estimada do nó a ser expandido até o destino final, escolhendo sempre o mais próximo do fim. Por fim o A*, assume que sempre encontrará o caminho até o destino final, com o menor custo possível, caracterizando seu comportamento como ótimo e completo, proposto por HART et al. (1968), esse algoritmo é diretamente influenciado pela heurística aplicada ao código, que quanto maior a eficiência desta, melhor será o desempenho do código. Seu comportamento baseia-se especificamente na avaliação dos vértices, combinando o custo para alcançar cada vértice e o custo para ir do vértice atual para o objetivo, a função de cada vértice é dada por $f(n) = g(n) + h(n)$, onde $g(n)$ é o custo do caminho do vértice inicial até o vértice atual n e $h(n)$ é o custo estimado do vértice atual até o objetivo, assemelhando $h(n)$ ao algoritmo guloso.

III. Navegação utilizando Controle Inteligente

Nesta seção será apresentado o desenvolvimento do software de navegação utilizando controles inteligentes que, através de implementações de algoritmos de busca, possibilitará o controle do robô por códigos inteligentes. Inicialmente será apresentado o ambiente de simulação criado e, em seguida, a aplicação dos algoritmos de busca inteligente.

A. Mapa do Ambiente

O mapa ou ambiente simulado é semelhante a um labirinto, que para os testes iniciais foi composto por oito salas, porém com o objetivo de explorar o potencial dos algoritmos de busca o mapa foi ampliado para vinte e duas salas. O mapa é apresentado na Figura 2.

Para a modelagem computacional inicial do problema em questão utilizamos um grafo para a representação do mapa de salas (Figura 3).

Considerados neste projeto o conhecimento prévio de informações sobre salas e suas respectivas portas, sendo assim, foi desenvolvida uma estrutura contendo as coordenadas de cada porta e as salas as quais exista acesso. A estrutura das salas, além de possuir o nome, possuem também as coordenadas de todas as extremidades, pois são necessárias para a verificação se o robô mudou de sala. Também é importante as informações sobre a posição central da sala, ou seja, as coordenadas, então quando o robô muda de sala,

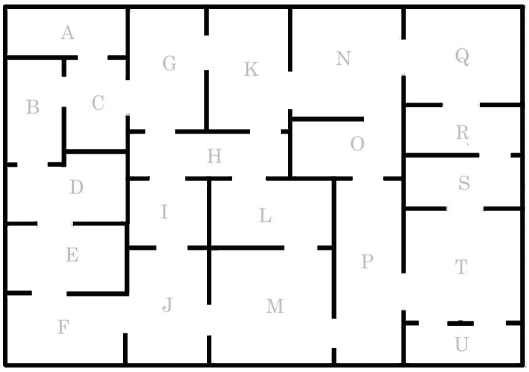

Figura 2. Mapa de salas do ambiente simulado.

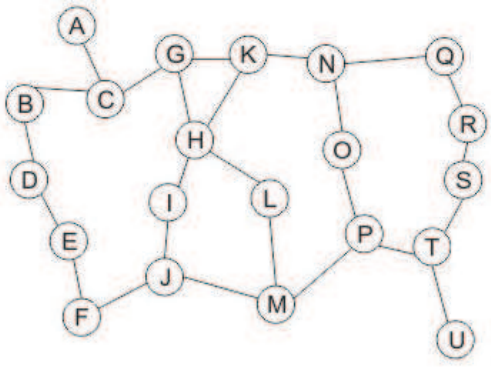

Figura 3. Grafo do mapa de salas.

o mesmo é direcionado para o centro da sala (evitando que fique a todo o momento direcionado para o objetivo, consequentemente encontrando obstáculos a todo momento), ao alcançar a posição cental o robô é direcionado para a porta da próxima sala e caso seja a sala do objetivo a execução é encerrada.

B. Software Desenvolvido

Embora o Player/Stage permita trabalhar com linguagens como Java e Python, C++ nos pareceu mais eficiente neste caso, pelo seu grande poder de comunicação com o hardware e consequente rapidez, sendo assim escolhemos a linguagem C++ para o desenvolvimento da solução para o problema proposto.

Na tela principal do aplicativo desenvolvido (Figura 4) poderá ser escolhido o algoritmo de busca a ser utilizado para a resolução do problema, neste caso a navegação autônoma, partindo de uma sala inicial, chegando em uma sala objetivo. De maneira geral, os principais métodos desenvolvidos foram:

- *wander*: tem como objetivo carregar todas as informações necessárias para o funcionamento da aplicação;
- *chooseDoor*: fazer o robô se movimentar e retornar a porta correta, dada a sala para a qual o robô deseja ir;
- *goToRoom*: direcionar o robô para uma determinada coordenada, podendo ser as coordenadas da porta escolhida no método anterior ou as coordenadas do centro da sala;
- *changeRoom*: verificar se o robô mudou de sala e, caso positivo, o mesmo é direcionado ao centro da nova sala.

Ao clicar no botão Iniciar (Figura 4) o aplicativo carrega todas as informações a respeito do ambiente, especificamente as localizações das salas e suas respectivas portas. Após isso o algoritmo de busca escolhido é executado, gerando como resultado a sequência de salas que o robô deve percorrer para chegar à sala objetivo partindo da sala origem, ambas escolhidas na tela inicial do aplicativo. Ao fim da execução do algoritmo de busca o robô começa a se locomover pelo ambiente, uma vez que este já o conhece e já 'decidiu' que caminho irá percorrer.

O processo de locomoção dá-se da seguinte forma: percebendo em que sala está e sabendo para onde deve ir, o robô passa a locomover-se para a porta que dá acesso à próxima sala da sequência que o algoritmo encontrou, ele continuará fazendo isso até que perceba que mudou de sala. Ao perceber a mudança de sala ele caminhará em direção ao centro da sala por cerca de 1 segundo, isso impede que ele fique todo o tempo direcionado para o objetivo, encontrando desnecessariamente obstáculos, no caso as paredes da sala, após esse período ele reiniciará o processo. Essa sequência de ações segue em *loop* até que o robô se encontre na sala objetivo.

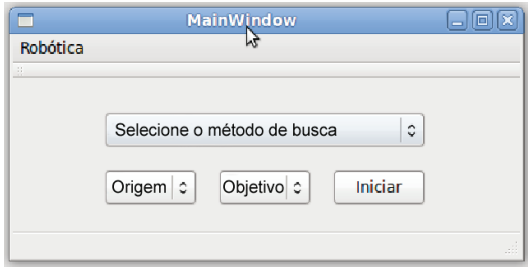

Figura 4. Tela inicial da aplicação.

Foram utilizadas também três estruturas de dados, sendo uma Fila, uma Pilha e uma Lista, necessárias a partir da particularidade de cada estratégia de busca. Os algoritmos de busca em largura, busca de custo uniforme e busca gulosa pela melhor escolha utilizaram a Fila como estrutura, a busca em profundidade fez uso da Pilha, pelo fato de necessitar lembrar-se para onde deve ir ao atingir um ponto sem saída e a busca A* utilizou a Lista, por facilitar a inserção e remoção de itens a todo o momento.

C. Resultados Obtidos

Para cumprir o objetivo deste trabalho foram implementados alguns algoritmos de busca inteligente. Pode-se verificar nos testes realizados que, quatro dos cinco algoritmos

de busca encontraram o melhor caminho para o objetivo. O algoritmo de busca em profundidade encontrou o pior caminho, comparado aos demais, o algoritmo em largura, embora tenha encontrado um caminho ótimo, necessitou expandir uma quantidade considerável de vértices, o que gerou maior tempo de processamento e uso de recursos como, por exemplo, memória, porém estas são características comuns dos métodos de busca sem informação.

Os algoritmos de busca gulosa pela melhor escolha e o de busca de custo uniforme encontraram caminhos melhores do que os algoritmos mencionados, entretanto o algoritmo de busca A* foi o único que garantiu o melhor caminho possível, pois, apesar de expandir mais nós que o algoritmo guloso, considera informações relacionadas ao objetivo e aos vizinhos locais.

Avaliando os custos para encontrar o caminho da origem até o objetivo definidos para os testes, notamos que os algoritmos de busca em profundidade, busca em largura e custo uniforme precisaram expandir 50 nós para encontrar um resultado satisfatório, sendo que nenhum deles pode garantir que o caminho encontrado seja o melhor que existe, apesar disso, neste caso específico, a busca em largura e a busca do custo uniforme encontraram o melhor caminho existente.

O algoritmo de busca gulosa foi o mais eficiente neste caso, encontrou o melhor caminho existente expandindo somente 23 nós, no entanto, assim como os citados anteriormente, ele não pode nos garantir o melhor resultado possível em virtude da natureza incompleta da sua heurística, que leva em conta somente o custo do nó a ser expandido até o destino final. Em contraste, o A* garante que o resultado encontrado será o melhor possível, pois considera a mesma heurística da busca gulosa combinada com o custo do nó atual até o próximo nó a ser expandido, no caso demonstrado o algoritmo A* apresentou um resultado intermediário, expandindo 39 nós e encontrando um caminho ótimo.

A Figura 5 apresenta as trajetórias encontradas pelos algoritmos de busca no ambiente apresentado na Figura 2.

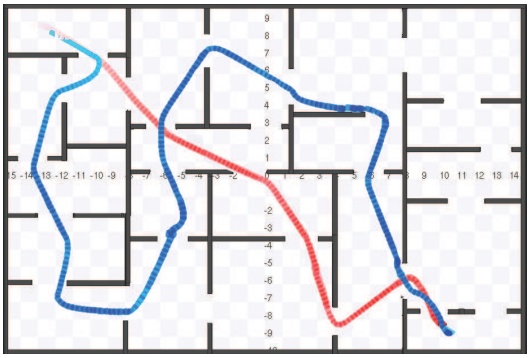

Figura 5. Trajetórias realizadas pelo robô.

A trajetória em *azul* foi o caminho encontrado pela Busca em Profundidade, já a trajetória em *vermelho* foi o caminho encontrado por todos os outros algoritmos que, nas condições propostas, encontraram o melhor caminho possível.

IV. Conclusões

Este trabalho teve como objetivo guiar um robô inserido em ambiente simulado, entender seu funcionamento e dar-lhe a capacidade de navegação autônoma.

Consideramos que os objetivos do trabalho foram alcançados, uma vez que, como resultados tivemos um aplicativo que simula um robô locomovendo-se de forma autônoma, através da aplicação de diversos algoritmos de inteligência artificial.

Como trabalhos futuros pretendemos ampliar o estudo para que o robô possa fazer o reconhecimento do ambiente de forma autônoma, comparando dessa forma a eficiência da locomoção do robô utilizando os métodos apresentados neste trabalho com a efeciência da locomoção do robô utilizando métodos de aprendizado. Isso nos proporcionará uma gama maior de trabalhos futuros, como por exemplo trabalhar com mais de um robô no mesmo ambiente.

Referências

HART, P., N. NILSSON, and B. RAPHAEL (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics 4*, 100–107.

LUGER, G. F. (2004). *Inteligência Artificial: Estruturas e Estratégias para a Solução de Problemas Complexos*. 4ł ed. Porto Alegre: Bookman.

OWEN, J. (2010). How to use player/stage 2nd edition. *The Player/Stage Manual*.

RIBEIRO, C., A. REALI, and R. ROMERO (2001). Robôs móveis inteligentes: Princípios e técnicas. *I Jornada de Atualização em Inteligência Artificial - JAIA2001, Anais do XXI Congresso da SBC 3*, 257–306.

RUSSELL, Stuarta e NORVIG, P. (2003). *Artificial Intelligence: A Modern Approach*. 2ł ed. Upper Saddle River, New Jersey: Prentice Hall.

WOLF, Denis F.and OSÓRIO, F. S. S. E. and O. Trindade Jr. (2009). Robótica inteligente: Da simulação às aplicações no mundo real. in: André ponce de leon f. de carvalho; tomasz kowaltowski. (org.). *JAI: Jornada de Atualização em Informática da SBC. Rio de Janeiro: SBC - Editora da PUC Rio 1*, 279–330.